Chandan Kumar Mishra
301280815, MSc CS -741 Non-thesis

Experiment:
I followed below steps to accomplish given data mining project.

Tools Used : Python, Eclipse, R console

1. Data Preprocessing : From given DocumentWords.txt and Vocabulary.txt , I converted it into form of Matrix size [301 * 7809] (where 301 is number of total documents and 7808 is number of total words from Vocublary.txt + 1 column for document Id). I observed that given data are sparse, so any available standard clustering tools such as Weka or R will not give consistent and proper clustering result.
To cope with this problem, first I reduced dimension of matrix by eliminating words which are having very less frequency(word which came in very few documents, taken threshold of 2). This has reduced dimension of matrix to approximatly half [301*2741] and after that by applying K-means in R console I am able to cluster documents in 4 different sets cross ponding to each mini assignment.

2. Clustering : After dimension reduction and by applying K-means using R, I was getting accuracy of approx 96 % (Tested against given validation jar). However, further I thought to improve clustering accuracy by using Semi-supervised learning by making use of information given in Collaboration.txt.
For this, I implemented modified K-means Algorithm in Python. In my modified K-means implementation, along with Ecludien distance to calculate simalirity between two documents, I am making use of information given in Collaboration.txt, as ons student Id should not come twice in same cluster.
By using this analogy, I improved an accuracy of clustering from 96.5 % to **99.64%** (validated using given validation jar).

3. Association Rule Mining: Once I had clustered documents crossponding to each Mini-Assignment, I used Apriori algorithm in R console to mine association rules. I tuned support and confidence parameter and finally used support= 0.15 and Confidence =0.8. This has given me set of association rules cross ponding to each cluster. Here in R console, I sorted this rules in descending order of support value and exported to .csv file.

4. Sorting Association Rule having same support by length: After that, I am organizing association rules in a fashion where rules having same support and maximal length will come first (Suppose two association rules rule#1 and rule#2 having same support and len-1 and len-2 respectively. My python implementation will sort rules such that rule#2 will be before rule#1). This will be useful because order of predicted word matters.

5. Prediction : Starting with doc#id1 and set of association rules from doc#1's cluster, I will go through each sorted association rules and match for set of words in selected doc#1, if words in LHS of rule is mathcing with words in particular document I will recommend RHS of association rules as probable missing word. This process will go till 5 missing words has been identified for doc#1.
Similarly, above process will be repeated for each document and finally I will get clusterwise document id's having 5 missing word.

6 . Data Postprocessing : In this step, clubing prediction result of 4 cluster in single file **#studentid.txt** to submit result. Tested against given Validation.jar and getting accuracy of approximately 36.40 %.

7. Result :

Clustering Accuracy	**: 0.9964**
Average MAP@5	**: 0.3640**

Analysis of the experimental results : Efficient clustering and association rule generation(sorted by maximal length rule for same support) are major part of this experiment.

Clustering Analysis:

After dimension reduction and feeding into standard algorithm like K-means or EM, I was getting clustering result of approximately ≈ 95-96% accuracy.

Here, key thing is making use of hint given in Collabaration.txt which give information about (U,V,D), for any student U and V who is part of cluster 1 should not be again part of cluster 1. Simialrly, this constarint will be applicable for all students. Finally, all cluster will be of almost equal size.

In my modifed K-means Implementation where I am restricting same student id to not fall in same cluster by making use of collabration information and getting below clusters. I ran with different seed value and this is best result I got having accuracy of ≈ **99.64 %**.

Cluster	Num of documents
1	73
2	74
3	73
4	81

Table 1: Clustering Result

Association Rule Generation Analysis :

My approach to sort association rules having same support by length is illustrated in Table 2 and Table 3. This has improved prediction performance by approx 4 %. In early stage of project, I was only sorting rules by support and scanning in document to predict missing 5 words which was giving accuracy of approx ~ 31-32 %. However, after sorting association rules by length also, I am getting accuracy of 36.40 %. And, this seems quite obvious to sort by length as while predict we should start with longest match to get more confidence in prediction.

Association Rules	Support
{18743} => {91519}	0.481481481
{18743} => {96675}	0.481481481
{76320,96675} => {82848}	0.481481481
{76320,82848} => {96675}	0.481481481

Table 2: Sample Association rule

Association Rules	Support
{76320,96675} => {82848}	0.481481481
{76320,82848} => {96675}	0.481481481
{18743} => {91519}	0.481481481
{18743} => {96675}	0.481481481

Table 3: Assocition Rule Sorted by len having same support

CrossValidation :

Also, to confirm my result wheather it is overfiting given data, I done 10-fold cross validation and below is result :

Run1	Run 2	Run3	Run4	Run 5	Run 6	Run 7	Run 8	Run9	Run10
34.83	31.66	31.16	34.59	33.36	31.35	34.15	33.13	31.07	32.32

Table 4 : Cross validation Result

Overall CV Average Accuracy = $\frac{1}{10}[\frac{327.61}{9} * 10]$ = **36.40 %**

References :

1. *Data Mining Concepts and Techniques*- By Jiawei Han,Micheline Kamber,Jian Pei **3rd** edition